Giving control over email to the recipientGiving control over email to the recipient
Message Distribution Protocol
Steve Atkins, CTO, Word to the Wise
steve@word-to-the-wise.com
Many of the current problems with email, both physical issues such as the amount of spam in recipients mailboxes or the legitimate email deleted by over-aggressive spam filters and social or emotional issues such as the frustration of many recipients with misuse of their email address or their uneasiness with giving out their email address due to fear of misuse, are due to current email infrastructure putting all control in the hands of the sender. This document explores some of the benefits of giving that control back to the recipient. Rather than focusing on stopping email delivery, as many proposed solutions to spam do, it concentrates on giving the recipient control over the delivery of wanted bulk email, while avoiding many of the problems of traditional bulk email delivery. This is a very important subset of email to consider, as it is this sort of message that makes up the vast majority of email wrongly discarded by spam filters.

While this document is just an initial discussion of the concepts, the protocol proposed is simple enough that an initial version could be deployed and in use within days.

Problems with wanted bulk email

Problems for the recipient

Once the recipient subscribes to a list they lose all control over the use of their email address. Even when the recipient unsubscribes from the list it's not unusual for a sender to resubscribe them at some point in the future. Senders often maintain multiple lists and subscribing adds the user to all those lists. Unsubscribing from all of them can be a slow, painful process.

The sender may (and often does) sell or trade the recipients email address to others, making it impossible for the recipient to stop the incoming email.

Every mailing list requires different interactions to subscribe, unsusbscribe, suspend mail, change options. They all work in different ways, and they all need to be dealt with independently.

Moving ISPs means a lot of work to move mailing list subscriptions, and some will be left behind - there's no equivalent of a change-of-address form.

Problems for the receiving ISP

Bulk email is usually sent badly, consuming lots of resources (for instance, many simultaneous connections to the ISPs incoming mailserver). This leads to overloaded mailservers, risking delay of one-to-one email. This needs to be managed, but communicating mailing policies to every bulk emailer is a large overhead, and bouncing mail sent in this way leads to customer complaints and more support overhead.

Bulk email is often rejected by spam filters, leading to customers losing wanted email and support overhead. It also makes the ISP appear unreliable, damaging customer confidence.

Due to the requirements to handle bounces correctly, most bulk email is sent one at a time to each recipient (to take advantage of VERP). In most cases the same email message is sent to each recipient, so the amount of email handled could be drastically reduced if only a single copy were sent.

Problems for the sender

Blocked, or routed to the 'spam folder', because it is sent in bulk.

Blocked, or routed to the 'spam folder', because it 'looks like spam', for instance it includes an unsubscription link.

Potential recipients fear signing up for mailing lists, as they expect that their email address will be sold or traded, leading to more spam.

Potential recipients fear signing up for lists as they often find it difficult to unsubscribe. This leads to reduced subscription rates, particularly for users who may want to subscribe briefly, to see if they like the list.

Tuning the format in which a message is sent so that all recipients can read it is difficult - because there's not usually any easy way for the recipient to choose whether they prefer HTML or plain text email, as one example.

Expensive to send, as a lot of infrastructure is needed to send it correctly.

Bounce-handling is complex - too aggressive and you lose recipients, too

conservative and you risk being blacklisted.

Much bulk mail is silently discarded, either at an ISP level or by being routed to a 'spam folder' where it is deleted unread. That makes it impossible to gather reliable readership figures - and also makes it very hard to fix the problem (if any) leading to it being discarded.

Closed-loop opt-in is the only easy way to avoid all the problems of falsified subscriptions, but leads to loss of a significant fraction of people attempting to sign up.

Moving an email address list from one outsourced mailer to another is painful and expensive, as there's no easy way to transfer the permission status of the list.

Email address churn usually leads to loss of subscribers as they move ISPs.

A Solution

Many of these problems can be avoided or reduced by transfering control over the subscription and delivery process to the recipient (or the recipients ISP) and moving delivery from SMTP to another protocol, better suited to broadcasting permission-based messages to multiple recipients.

This proposal doesn't attempt to Solve All The Problems With Email, concentrating rather on the issues with wanted bulk email. The approach is one which can be deployed cheaply, easily and gradually, such that a mailing list can be distributed both via email and via this technique simultaneously. It can also be adopted by end-users (senders or recipients) with no deployment needed by ISPs, or it can be deployed by recipients ISPs to gain further benefits, including near-complete transparency to the end-user recipient.

A working name for the proposed solution is "Message Distribution Protocol", or MDP.

Goals

There are a number of goals for this proposal.

De-facto standards-based. Based on widely-used existing standards and protocols where possible, allowing reuse of current code, applications and protocol infrastructure. De-facto standards that exist and are in active use are preferred over unused standards, even if the latter have better engineering aesthetics.

No flag-day needed to change from the current system of bulk email to MDP - this system can be grafted on to current mail delivery methods (with varying degrees of transparency) and run in parallel.

Recipient retains control over subscriptions - only the recipient can add and remove themselves from a distribution list.

All distribution lists a user is subscribed to via MDP are managed in a uniform manner - the user can subscribe, unsubscribe, suspend or resume delivery of messages, or move delivery to another ISP easily.

Recipients email address is not passed to the sender in any manner, eliminating the risk of senders misusing it (and so, eliminating the recipients fear of them doing so). But a unique token (either unique per recipient, or unique per recipient/sender pair) is passed to the sender, allowing the sender a true recipient count.

Recipient can set delivery preferences (prefer HTML vs plain-text, PDA format, preferred languages etc) and pass them transparently to the sender.

A single copy of a message can be sent to an ISP, then exploded out to all the recipients at that ISP, saving bandwidth and storage needs. However, the list of recipients tokens is still passed back to the sender, maintaining the true recipient count.

Deliveries are initiated by the recipient, allowing ISPs to schedule deliveries at times of low system load and for recipients to fetch all outstanding deliveries when connected to the network (for instance, directly to a PDA when the PDA is synchronised, or to a laptop at a wireless hotspot).

It is possible for senders to publish delivery schedules to recipients, so recipients will only attempt to fetch on a reasonable schedule. Conversely, for urgent distribution lists with an unpredictable schedule, there is the ability to push a delivery (via some form of asynchronous notification) as soon as a message is available.

Recipients may be required to authenticate with senders, allowing distribution of valuable content via MDP.

Simplicity. It must be easy to write code to send or receive MDP messages, and possible in most cases to send or receive them by hand.

To get people to consider possibilities other than authentication and filtering when looking at the problems of spam, and to consider solutions that are appropriate for part of the email domain, rather than all of it.

MDP Specification

IMPORTANT: This 'specification' hasn't been spell-checked, let alone implemented. The aim is for the reader to take a few moments to think about the concepts concerned, and to possibly start some discussion. It's not to persuade people that this is The Answer and it Must Be Implemented Now, nor to nitpick about implementation details.

The closest application protocol to offering this currently is RSS2 (Really Simple Syndication) an XML based protocol for syndicating HTML content across multiple websites. Experimental software for gatewaying mailing lists to RSS streams is already available and in use, as are end-user applications to view RSS streams as email.

RSS is a stable protocol, so code implementing it is not chasing a moving target. It has no known intellectual property issues. It is usually layered over a widely supported transport protocol, such as http or http/ssl, allowing it to be easily used through current firewalls, proxies and NATs.

The requirements of MDP and RSS are not close enough that MDP can be cleanly implemented as an extension of RSS, but by basing MDP on RSS it allows much of the code already being used for RSS (both for publication and for reading) to be reused for MDP with minimal changes, and allows any MDP client to easily support subscription to RSS feeds as well. (It would also allow XSLT level translation between the two in many cases).

In its simplest configuration the sender of an MDP list needs only to place static text files on a webserver (probably with the aid of a GUI application or web script to generate the XML index file).

Initially a recipient may subscribe to an MDP list by pasting the subscription URL of the list into their local MDP client, or into the web interface to an MDP client provided by their ISP. Reading MDP list traffic might then be performed through the MDP client, or the MDP client may insert the messages into the recipients mailbox (equivalent to an SMTP local delivery agent).

An MDP-SUBSCRIBE document is a well-formed XML document, available via a recognised protocol, typically http or https.

At the top level an MDP-SUBSCRIBE document is a single <mdp> element, with a mandatory attribute called version, that specifies the version of MDP the document conforms to. If it conforms to this specification, the version attribute must be "0.1".

The <mdp> element contains one or more <channel> elements. <channel> has two required elements - <title> and <description>.

<title>
contains the name of the channel.
<description>
contains a brief description of the channel

<channel> has the following optional elements

<link>
contains the URL to the website corresponding to the channel - this is required if there is such a website.
<image>
Specifies a GIF, JPG or PNG image that can be displayed with the channel. It contains one required element, <url>, the URL of the image and two optional elements, <width> and <height> giving the width and height of the image in pixels.
<url>
contains the URL from which the MDP document that is the channel can be retrieved. It is optional, and if missing defaults to the URL that the MDP-SUBSCRIBE document was retrieved from. This means that in simple cases an MDP document can also be its own MDP-SUBSCRIBE document.
<user>
contains one required element, <id>.
<id>

contains an identifier that will be passed back to the sender on each request from this recipient.

Use of <user> is discouraged, as it can add significantly to overhead. It should only be used where authentication of recipients is needed.

(With web browser support - possibly under an application/mdp content type or similar - the recipient can click on a link in a webpage to popup a window describing the distribution list to them, and giving them an opportunity to subscribe. With a small amount of browser support this can support both local MDP clients and web-interfaces to ISP level MDP clients.)

An MDP feed is a well-formed XML document, available via an http (or https) POST or GET request. The URL is that given in the <url> section of the MDP-SUBSCRIBE document.

The following are passed as http parameters along with the request:

guid

A globally unique parameter. Whenever a given installation of an MDP client retrieves a particular MDP list it must send the same guid. For a server-side MDP client feeding multiple users it might be the machines hostname. For an individual recipients MDP client it might be a hash of the local part of their email address and a salt, combined with the domain part of their email address. There is no requirement that an MDP client use the same guid with all lists, only that it use it consistently with each list.

since

The RFC 2822 date-time of the last request this client made to this list. Only <items> after this date-time will be returned.

user

If the list requires per-user authentication then one or more user identifiers (as returned in the subscription document) are passed as one or more user= parameters.

count

If this list doesn't require per-user authentication then the count parameter contains the number of mailboxes the message will be delivered to.

(This is extremely lightweight, making it trivial to implement, but isn't very elegant. Something like SOAP would make it more elegant, perhaps.)

At the top level an MDP document is a single <mdp> element, with a mandatory attribute called version, that specifies the version of MDP the document conforms to. If it conforms to this specification, the version attribute must be "0.1".

The <mdp> element contains exactly one <channel> element containing zero or more <item> subelements.

<channel> has two required elements - <title> and <description>.

<title>

contains the name of the channel.

<description>

contains a brief description of the channel

<channel> has the following optional elements

<link>

contains a URL to the website corresponding to the channel - this is required if there is such a website.

<lastBuildDate>

contains the last time the content of the channel changed, as an RFC 2822 format date.

<cloud>

used for publish-subscribe asynchronous notification (see RSS 2.0 spec for three ways to handle this, but there are others).

<expires>

contains the date and time before which clients should not attempt to reload the document, as an RFC 2822 format date. There is an optional non-negative numeric attribute called 'jitter' - an MDP client should choose a random-ish number between zero and the value of jitter and add that number of seconds to the date in expires before use. A missing jitter is equivalent to a jitter of zero.

<ttl>

contains the number of seconds before which the client should not attempt to

reload the channel. A channel should not have both <ttl> and <expires> elements.

Why the choice of elements?

Any channel with an ill-defined posting schedule is likely to want to use <ttl> so all clients poll the channel regularly - unless it also uses <cloud> for asynchronous notification.

A statically generated channel with a regular posting schedule will probably want to use an <expires> element with a non-zero jitter to ensure that all subscribers receive the next message within a well-defined window, but they won't all request it at the same moment. Clock skew between server and client means that this will not always give the desired result.

A dynamically generated channel with a regular posting schedule can use <ttl> to suggest the client reload the channel at a specific time in the future. This avoids uncertainties due to clock skew between server and client.

A channel with neither element will not be reloaded by a client on any particular schedule, and may not be reloaded at all - unless the channel uses <cloud> for asynchronous notification.

<image>

Specifies a GIF, JPG or PNG image that can be displayed with the channel. It contains one required element, <url>, the URL of the image and two optional elements, <width> and <height> giving the width and height of the image in pixels.

<rating>

contains the PICS rating for the channel. If the MDP client uses this to restrict access, it is strongly suggested that access be denied to any MDP channel with no rating.

<archive>

contains a URL to another MDP document, containing the same distribution list as this MDP document, but containing a longer list of <item> elements.

<redirect>

specifies that the client should not use this <channel> element, but instead go to a URL elsewhere. It has one optional attribute, 'temporary'. If the attribute is set then the MDP client should not use this channel element, but instead immediately retrieve it from a new document at the contained URL this time only.

If the attribute is missing then the MDP client should use only the title and description from this channel, and should permanently replace the URL this channel was received from with the contents of the <redirect> tag.

This is used in two main cases.

A temporary redirect may be used for load-balancing or temporary load-shedding to a higher capacity server.

A permanent redirect will typically be used when the sender is moving from one provider to another. They can replace their old MDP document with a static document containing a single channel with a permanent redirect to their new URL. Note that this is the only case where an MDP client should follow a permanent redirect without prompting the user.

A <channel> element MUST NOT have both a <redirect> subelement and an <item> subelement.

<item>

the real payload of the MDP document. There is one <item> subelement for each recent message (where recent is defined later). An <item> contains a number of subelements. The <title> subelement is required, as is one or both of <link> or <description>

<title>

contains the title of the item, equivalent to the subject of an email.

<link>

contains a URL pointing to the content of the item. This is loosely equivalent to the body of an email. It has an optional parameter, 'type' specifying the MIME content-type. There may be multiple <link> elements, allowing the MDP client to choose which content type to use. As an extension to MIME content types, 'text/html/text' refers to a simple subset of HTML excluding images suitable for rich-text on small clients and 'text/html/simple' refers to a simple subset of HTML including images.

(Specify). (language parameter?).

<description>

contains a plain-text synopsis of the item. If there is no <link> subelement this may contain the entire content of the item.

<sender>

describes the sender of the document. It has one optional parameter, url, which gives a URL that can be used to directly contact the author of the item. The content is the descriptive name of the author. So the equivalent of an email

From: Steve Atkins <steve@blighty.com>

would be

<sender url="mailto:steve@blighty.com">Steve Atkins</sender>

The URL needn't be a mailto: URL - it could perfectly well be a web page or any other way of contacting (possibly indirectly) the sender.

<pubDate>

contains the date the item was distributed, in RFC 2822 format. It is directly equivalent to the Date header in email.

<comments>

contains the URL that may be used to respond to the item. If used with a mailto: URL it is directly equivalent to an email Reply-To: header.

<guid>

contains a globally unique identifier for the message. This has one optional parameter, isPermaLink, which defaults to the value "false".

If isPermaLink is "true" then the content of the element must be a URL which can be accessed at any point in the future to retrieve an archived copy of this message.

Otherwise, there are no constraints on the content of <guid> apart from a requirement it be globally unique - it is directly equivalent to an email Message-ID.

<user>

If an <item> contains one or more <user> subelements, it should be delivered to all of those recipients. If it contains no <user> subelements then it should be delivered to all recipients for whom the channel was requested.

(This document is available at http://word-to-the-wise.com/message_distribution)